Paper prototyping

Sure, it's low-tech, but this usability testing method can help you sidestep problems before you write your code

Carolyn Snyder (csnyder@snyderconsulting.net)
President, Snyder Consulting
November 2001

Wouldn't it be great to find out what users (and marketing) want *before* you start coding? Paper prototyping lets you do just that. While it may seem counterintuitive to test an interface without using a computer, paper prototyping lets you get maximum feedback for minimum effort. After a few usability tests with a paper prototype, you'll have confidence that you're implementing the right thing.

Contents:

What paper prototyping is
How good should the prototype be?
What paper prototyping is (and isn't) good for
Key benefits for designers and developers
Common concerns
Conclusion
Resources
About the author
Rate this article

Related content:

Seven tricks that Web designers don't know
Subscribe to the developerWorks newsletter
More dW Usability resources
More dW Web architecture resources

What paper prototyping is

Paper prototyping is a method of usability testing that is useful for Web sites, Web applications, and conventional software. Here's how it works: You first decide on the tasks that you'd like the user to accomplish. Next, you make screen shots and/or hand-sketched drafts of the windows, menus, dialog boxes, pages, popup messages, etc. that are needed to perform those tasks. Then you conduct a usability test by having one or two developers play the role of "computer," manipulating the pieces of paper to simulate how the interface would behave. Users are given realistic tasks to perform by interacting directly with the prototype -- they "click" by touching the prototype buttons or links and "type" by writing their data in the prototype's edit fields. (Using transparency or removable tape prevents the prototype from being written on directly.)

A facilitator (usually someone trained in usability) conducts the session while other members of the development team observe and take notes. The "computer" does not explain how the interface is supposed to work, but merely simulates what the interface would do. In this manner, you can identify which parts of the interface are self-explanatory and which parts are confusing. Because the prototype is all on paper, you can modify it very easily to fix the problems you find.

Figure 1. A paper prototype of the File Setup dialog from Microsoft Word

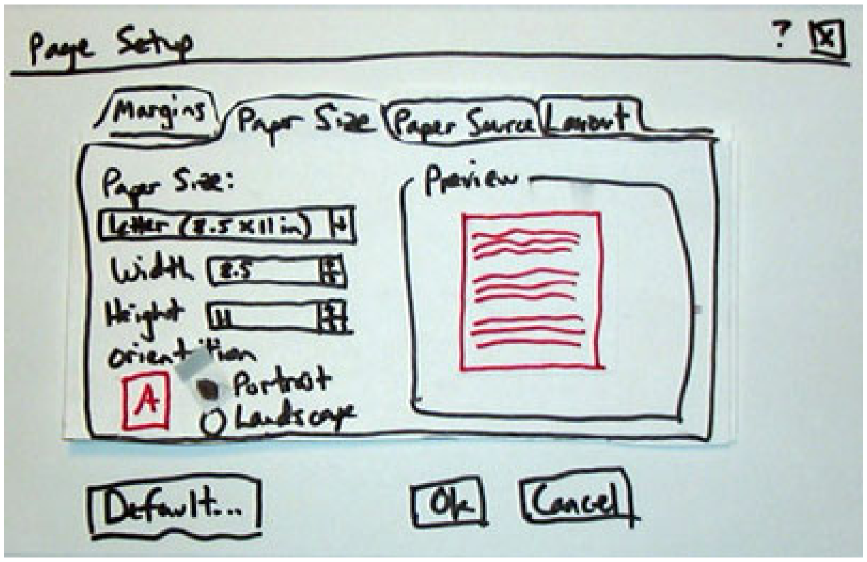
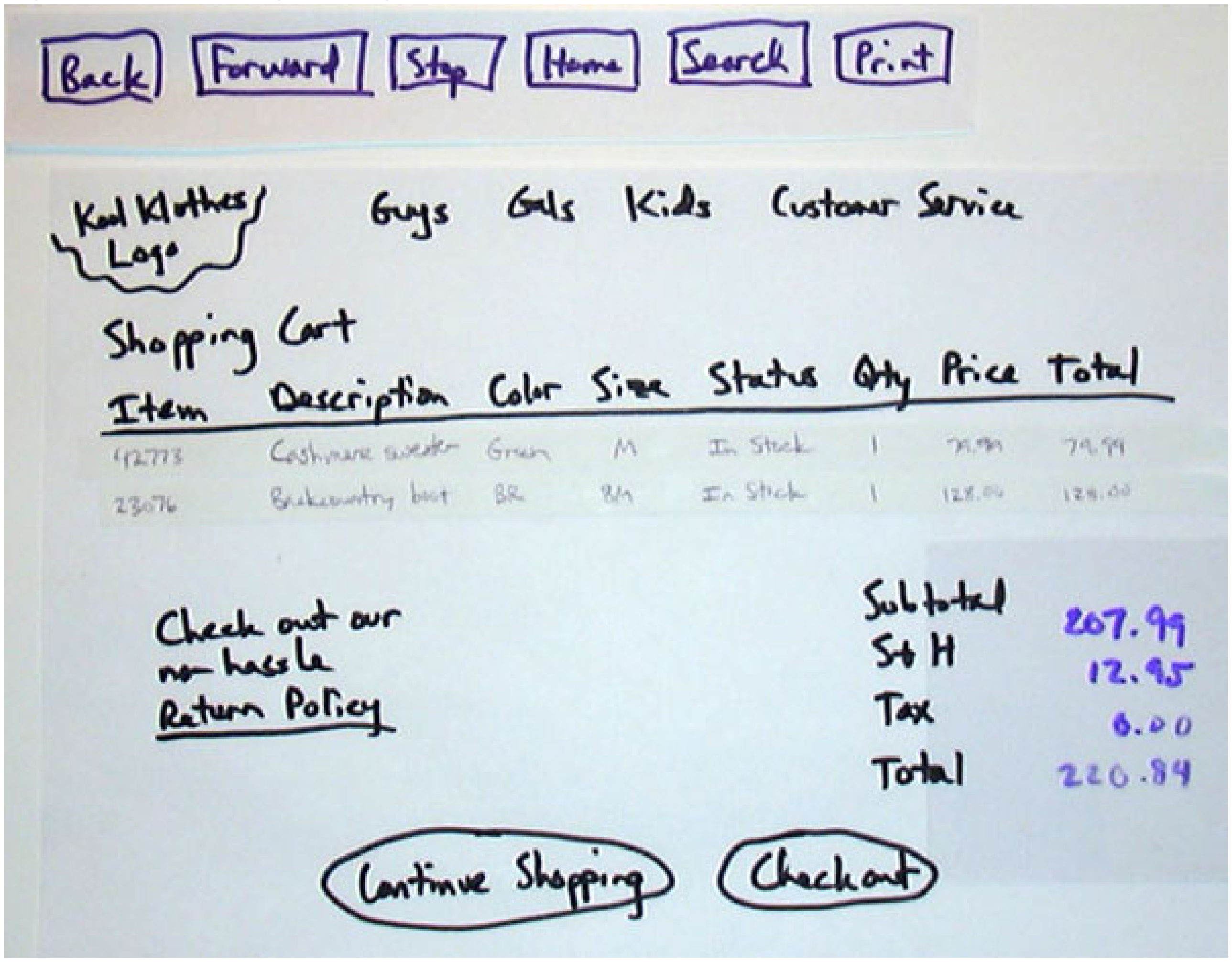


Figure 1 illustrates how various GUI widgets can be prototyped. This example shows the File Setup dialog from Microsoft Word. (In a usability test, this component would be placed on top of a prototype of the Word application rather than being shown by itself.) Each tab is on a separate piece of paper so it can be moved to the front if the user selects it. The drop-down list for paper size is written on a separate piece of paper and is shown if the user touches the drop-down arrow. The radio button is simulated using removable tape. The preview components (shown in red for purposes of this illustration) are tacked on with restickable glue so they can be changed if the user selects Landscape. (See Resources for more information on where to get removable tape and restickable glue.)

Figure 2. The shopping cart page from an e-commerce site



Item	Description	Color	Size	Status	Qty	Price	Total
42773	Cashmere sweater	Green	M	In Stock	1	79.99	79.99
23076	Backcountry boot	BR	8M	In Stock	1	128.00	128.00

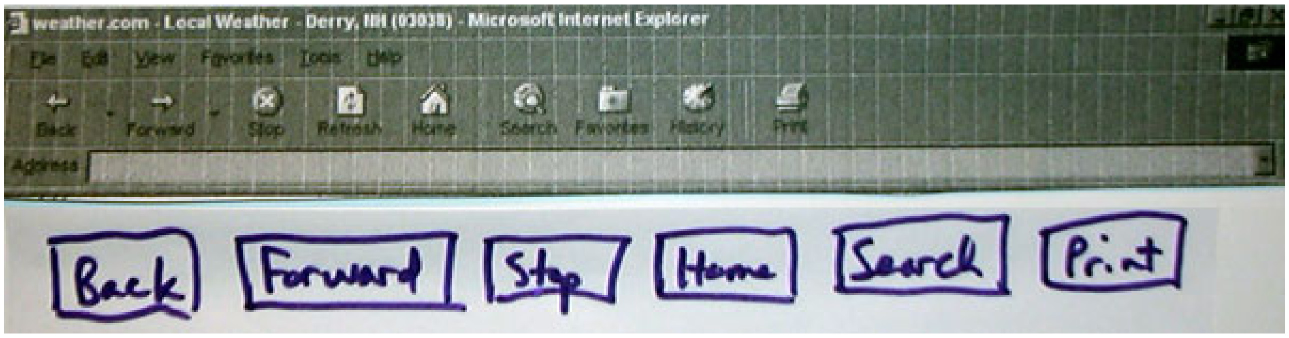
Subtotal	207.99
S+H	12.95
Tax	0.00
Total	220.84

Figure 2 shows how removable tape allows users to place any combination of items in the shopping cart.

The "computer" can update the shipping cost and total accordingly by wiping them off the transparency with a damp paper towel and rewriting them. (It's not considered cheating for the "computer" to use a calculator!)

How good should the prototype be?
You can use screen shots of an existing design if you happen to have them, but it's also fine to hand-sketch them, especially when you're in the early stages of design. And as Figure 3 shows, sometimes hand-drawn elements are actually more readable than screen shots that use a dark background color.

Figure 3. Hand-drawn versions of browser buttons compared to a grayscale screen shot


weather.com - Local Weather - Derry, NH (03038) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Print
Address
Back
Forward
Stop
Home
Search
Print

You can also mix and match screen shots and hand-drawn components. The prototype only needs to be good enough for you to get answers to the questions you're most concerned about. Thus, most paper prototypes don't need:

- **Straight lines or typed text.** If the user can't read something, it's OK to tell them what it says. (But if the user doesn't understand a term, don't explain it -- change it.)

- **Images or icons.** Use words instead. For example, for the company logo, you can just draw a box around the words "company logo." If images are part of the content (a product catalog, for example), you can paste them into your prototype using restickable glue, which allows you to rearrange the page later.

- **Color.** As the saying goes, "You can put earrings on a pig, but it's still a pig." Color can't save an inherently flawed design -- do your initial testing with grayscale printouts of screen shots, or sketches using any dark-colored marker. Color can be added later once you're sure you aren't creating a pig.

- **Consistent sizing of components.** Unless you've got a small or densely-packed display, don't worry about adhering exactly to a grid. It's OK if components are of varying sizes. For example, perhaps your menu bar and icons came from a screen shot and are relatively small compared to the hand-drawn parts of the screen. You can clarify by saying things like, "This is all one window," if the user seems confused.

It's fine if the prototype looks a bit messy. Very often, the first usability test will show you problems you'd never anticipated, and then you'll want to make changes. Don't spend time making the prototype look neat before you test it -- if it's legible, it's good enough. (For those of you who are thinking, "But I can easily mock up something that looks nice," see the sidebar.)

What paper prototyping is (and isn't) good for
Paper prototyping is especially useful for gathering data about the following kinds of problems:

- **Concepts and terminology.** Do the target users understand the terms you've chosen? Are there key concepts they gloss over or misconstrue? (I've seen these types of usability issues in virtually every interface I've ever tested.)

- **Navigation/workflow.** If there's a process or sequence of steps, does it match what users expect? Do they have to keep flipping back and forth between screens? Does the interface ask for inputs that users don't have, or don't want to enter?

- **Content.** Does the interface provide the right information for users to make decisions? Does it have extra information that they don't need, or that annoys them?

- **Page layout.** Although your scribbled screens may not be pretty, you'll still get a sense of whether users can find the information they need. Do you have the fields in the order that users expect? Is the amount of information overwhelming, not enough, or about right?

- **Functionality.** You may discover missing functionality that users need, or functionality you'd planned but users don't care about.

On the other hand, paper prototyping isn't ideal if your biggest questions pertain to:

- **Technical feasibility.** Paper prototypes don't demonstrate technical capability. It's possible to create a paper prototype that can't actually be implemented. To avoid this, I recommend that there always be at least one person involved who understands the technical constraints.

- **Download time or other response time.** Because a person simulates the behavior of the computer, the "response time" is artificial.

- **Scrolling.** I've seen some interesting and subtle problems with Web page designs that discourage the user from scrolling either down the page or back up to the top. I wouldn't have found these problems with a paper prototype.

- **Colors and fonts.** If you really need to see how something looks on a computer screen, paper prototyping can't show you that. It's a good idea to involve the graphic designer in the paper prototype tests because he may find issues that influence the visual aspects of the final design.

Some development teams use paper prototypes in the early stages to smoke out the show-stoppers, and then do a couple of additional usability tests later with the real interface just to look for any additional surprises.

Key benefits for designers and developers
If you design and/or implement user interfaces, paper prototyping can provide several benefits:

1. Test your design with users before you code
Paper prototypes are ideal for finding out whether you're on the right track, before you write even a line of code. Sometimes the things you learn can have a significant impact on the interface or even the underlying architecture. For example, in testing a prototype of a travel Web site, we learned that users insisted on seeing some information that had to come from the airlines' databases. On one hand, this was bad news because it was technically difficult to get this information. On the other hand, the team learned about this need in time to address it in the first release, or else the site would have failed.

2. Make fast changes
With a paper prototype, you can revise the interface very quickly, even during a usability test. If a term is confusing, you can cross it out and try a different one. You can add an example next to a confusing edit field or a sentence of explanation. Simple changes like these can solve many usability problems. Even when you find the need for more substantial changes, you can often make them in a matter of hours. One team completely redesigned an e-commerce interface in one evening after learning from the first two tests that it had some fundamental flaws. The next day's tests confirmed that they'd fixed the problems.

3. Find out what marketing really wants
It's frustrating to implement something that wasn't quite what your marketing department had in mind. I call this the "Bud Light problem" after a funny series of Budweiser commercials in which someone says, "Bring me a light!" and is then presented with a lamp, flaming torch, fireworks -- anything but a light beer. It's not so funny when you've spent days or months developing the wrong thing. Interface specs -- even with screen shots -- don't solve the Bud Light problem because they don't show the behavior of the system, and they also don't show you what people (especially non-technical people) expect the interface to do. But when you put a paper prototype in front of someone and watch them use it, you'll quickly discover whether it meets their expectations. Invite marketing to your prototyping sessions, and make sure they

attend the usability tests.

4. Eliminate technology variables from the usability testing equation
In eight years of usability testing, I've collected a number of "war stories" where technical glitches caused tests to be postponed or cancelled. This was disruptive to the development teams, which were relying on the tests to answer important questions about the interface. But I've never had to abort a usability test due to a problem with a paper prototype. I've even done paper prototyping during a power failure!

Paper prototypes give you complete control over how the interface behaves -- this is useful for short-notice demos as well as usability tests. Paper prototypes don't use databases, networks, servers, or any other form of technology, which eliminates most of the things that can go wrong. You don't have to worry about someone putting a change up on the development server that breaks the very thing you're trying to test (and this also means that development doesn't have to come to a complete stop every time you want to do a usability test).

You may be thinking, "But this is my application. I'm the one writing the code." Yes, but until you're well into the debugging stage, your code (or code written by others that yours depends on) may not be stable enough to allow users to do meaningful work without crashing. Bugs happen -- there's a big difference between the pleasant fantasy of "These ones and zeros obey my every whim" and the painful reality of "I'd be better off if I hadn't come in today." I hope I'm not the only software developer who's ever said that!

Common concerns
Even people who recognize the benefits of paper prototyping may have some concerns about the technique. The questions I commonly hear fall into three categories: validity, professionalism, and resources.

1. Validity ("Does it find real problems?")
I'm often asked whether paper prototyping might have bad side effects, either in terms of missing important problems or finding false ones. The short answer is that paper prototyping does find most of the same problems that you'd find by testing the real thing.

To my knowledge, there is only one published scientific study (see Resources) of the validity of paper prototyping. Three researchers at Verizon (formerly GTE Labs) compared the type and number of problems found with a low-fidelity (i.e., paper) prototype as compared to a working prototype. They found that there was a significant degree of overlap between the problems found using the two different methods. Although the working prototypes did uncover a few more problems, they also took significantly longer to develop than the low-fidelity ones -- weeks instead of days. These results indicate that there are diminishing returns from taking the additional time to develop a polished prototype.

My own practical experience backs this up. I have conducted several hundred usability tests of software and Web sites, including more than 100 using paper prototypes. With only a few exceptions, I find the same kinds of problems with both.

2. Professionalism ("What will others think of it ... and us?")
The thought of showing an unfinished -- or even flawed -- design to outsiders makes some developers uncomfortable. Many of us are perfectionists by nature and/or training, and our jobs reward this trait. It's natural to be concerned that others will perceive our work as incomplete or sloppy.

Fortunately, that's not what happens. Users (and coworkers) respond positively to a paper prototype, provided that they've been briefed appropriately. When participants arrive for a usability test, I explain that we know the design has some rough edges (sometimes literally!) but that we want to get their feedback before we've invested a lot of effort to build the wrong thing. I've found that users understand this, and I've never heard

Why not just use Visual Basic, HTML, etc.?

There are some inherent advantages that paper prototyping offers over any computer-based prototype, no matter how proficient you are with the tool.

1. *Zero coding effort.* While it's possible to mock up a decent-looking interface pretty quickly in VB, Dreamweaver, etc., writing the code to make the interface respond properly to the user's inputs can be time-consuming. With a paper prototype, the time spent coding is zero -- even if you're a real whiz, it's hard to be faster than that!

2. *Avoid nit-picky feedback.* A polished-looking design can actually encourage the wrong kind of feedback. If you're trying to make sure you've got the right content and functionality, you may not want to hear

a user make a disparaging remark about a paper prototype. On the contrary -- users appreciate knowing that their input is being requested while there's still time to incorporate it.

3. Development resources ("Do we really have time to do this? Is it just extra work?")
Those who have never done paper prototyping (or even usability testing) often overestimate the amount of time it will take. If you're already doing usability testing and only the paper prototype part is new, estimate no more than 1-2 days for prototype creation if you're working mostly from screen shots, and up to a week if it's a new interface or a complete redesign. Any more than a week and you're spending too much time polishing the design before getting user feedback. (The other activities associated with planning, preparing, and conducting a handful of usability tests typically take on the order of a week of hands-on time, spread out over 2-4 weeks.)

Sometimes paper prototyping will uncover planned functionality that doesn't need to be implemented. For example, I once tested a paper prototype of a distance learning Web application. The original design called for users to have avatars to navigate the virtual classroom, raise their virtual hands, etc. In prototype testing, we quickly learned that all this virtual 3D stuff got in the way of the real purpose of the application, which was to enable large companies to provide training to geographically dispersed employees. The designers realized that they were jeopardizing their whole product concept by trying to make the application feel like a video game. They dropped the 3D functionality, which actually shortened their development schedule. This was five years ago, and the product is still around, sans 3D.

comments like, "Hey, those fields don't line up," or "I don't like that shade of green." Paper prototypes avoid that kind of feedback because it's obvious to users that you haven't specified the look yet. This encourages users to focus on the concepts and functionality.

3. *Encourage creativity.* Our brains respond more creatively to things that look somewhat unfinished. And users -- especially non-technical ones -- are often less intimidated by paper prototypes than by computers, so they'll feel more comfortable exploring your design.

Conclusion
It's normal for interfaces to undergo substantial changes between the initial concept and the actual release. No one enjoys writing code that ends up having to be overhauled or scrapped. Paper prototyping lets you find significant problems early in the design process, so you can experiment to find the best solution before investing the effort to implement it.

Resources

- The article "Usability problem identification using both low- and high-fidelity prototypes" by Robert Virzi, Jeff Sokolov, and Demetrios Karis is available from the ACM Digital Library. (You must be an ACM member or pay a US$10 fee to download individual articles.)
- Read the author's list of useful office supplies for paper prototyping, including hard-to-find items like removable tape. Includes links to specific items on two office supply Web sites.
- The book *A Practical Guide to Usability Testing* by Joe Dumas and Ginny Redish is a good introductory reference for usability testing.
- The Society for Technical Communication maintains a "Topics in Usability" site containing resources on usability testing.
- From User Interface Engineering's site, see the articles "Five Paper Prototype Tips" by Matthew Klee, and "Using Paper Prototypes to Manage Risk" by Carolyn Snyder.
- Check out the developerWorks article, "Seven tricks that Web designers don't know," also by Carolyn Snyder, to find out what you may be taking for granted about your users.

- IBM's Ease of Use site offers new innovations, user-centered design, guidelines, stories, technologies, and other resources to help improve the total user experience for your products and services.

- And don't forget accessibility. IBM's Accessibility Center is designed to provide people with disabilities greater access to information and technology.

About the author

Carolyn Snyder spent the first 10 years of her career as a developer of unusable software. Upon discovering users, she became a usability consultant, spending six years at user interface engineering before starting Snyder Consulting in 1999. She specializes in paper prototyping and usability testing. She's writing a book on paper prototyping and welcomes questions. E-mail her at csnyder@snyderconsulting.net.

PDF e-mail it!

What do you think of this article?

○ Killer! (5) ○ Good stuff (4) ○ So-so; not bad (3) ○ Needs work (2) ○ Lame! (1)

Comments?

Submit feedback